

# FathGrid

developer documentation

[www.fathsoft.com](http://www.fathsoft.com)

## Contents

Installation.....	1
Initialization .....	1
Configuration options.....	2
Column configuration options.....	4
Grid data.....	4
Inline data.....	5
Dynamic data.....	5
Server-side data.....	5
API .....	6
How to.....	7
Display calculated columns .....	7
Group by expression.....	7
Add column footer .....	7
Change appearance of some rows .....	8
Editing.....	8
Insert new row .....	8
Delete row .....	8
Show sub-grid.....	8

## Installation

Just add fathgrid.js to your page HEAD section

```
<script src="fathgrid.js"></script>
```

## Initialization

First, make sure you have a table with an ID attribute in your HTML:

```
<table id="table1" class="table table-hover table-bordered ">
  <thead class="thead-light"></thead>
  <tbody></tbody>
</table>
```

To initialize FathGrid object in place of a table, use JavaScript code:

```
var t1=FathGrid("table1",{ ...configuration_options });
```

## Configuration options

Configuration options is an object set as second parameter to FathGrid initialization call. It can contain the following fields:

<b>Field</b>	<b>Description</b>	<b>Default</b>
<b>size</b>	page size	10
<b>page</b>	page number	1
<b>filterable</b>	allow filters in table header	true
<b>editable</b>	allow editing when clicking on cells	false
<b>pageable</b>	allow paginator	true
<b>sortable</b>	allow sorting by clicking on column headers	true
<b>resizable</b>	allows interactive column resizing	true
<b>restoreColumns</b>	if true, automatically restore column widths from previous session	false
<b>showFooter</b>	show table footer	false
<b>showGroupFooter</b>	show group footer	false
<b>data</b>	data array	[]
<b>rowClass</b>	function (item, idx) which returns string of class names to add to each TR element in table body	
<b>onChange</b>	function to be called whenever cell data is changed	function(item, col, oldval, newval)
<b>groupOn</b>	function (item, index) which returns grouping string	function(item. idx){}

<b>sortBy</b>	array of column indices for fixed filter	[]
<b>columns</b>	array of column configuration objects	[]
<b>serverURL</b>	templated URL to retrieve JSON data from	null
<b>prepareData</b>	custom async function which converts received JSON object into data array	async (json)=> json
<b>loading</b>	message to show while loading data	'Loading...'
<b>rtl</b>	set to true for right-to-left languages	false
<b>selectColumns</b>	allow hiding/showing columns	false
<b>onInitFilter</b>	function which is called when filter row is created. Useful for customizing appearance of filter controls	function(tr_element){}
<b>onInitTable</b>	function which is called when table body is rendered. Useful for customizing appearance	function(tbody_element){}
<b>onInitInput</b>	function called after table cell editor is initialized. Function parameters are: - <b>item</b> : data item - <b>idx</b> : column index - <b>el</b> : TD cell element in which input is located	function(data_item, column_index, TDElement){}
<b>template</b>	Templated HTML string for grid wrapper element. Use this to insert custom HTML between grid elements.	{tools}{info}{graph}{table}{pager}
<b>graphType</b>	graph type - line, bar, pie, etc.	line
<b>graphValues</b>	function which returns an object to initialize data graph	{ title: 'Graph Title', labels: ['Jan','Feb','Mar',...], values: [100,200,300,...] }

## Column configuration options

Column configuration object can contain the following fields:

<b>Field</b>	<b>Description</b>	<b>Default</b>
<b>name</b>	field name	
<b>visible</b>	if false, hide this column	true
<b>filter</b>	list-of-values for filter or <b>null</b> to automatically create list from table data	undefined
<b>value</b>	function(item) which returns text to display in column	
<b>width</b>	width in pixel units, if value is number. To use other units, use string value with units suffix eg. '10rem'.	auto
<b>html</b>	function(item) which return HTML for column cell	
<b>header</b>	text to show in column header	
<b>footer</b>	function(items, element) which returns text to display in footer cell	
<b>groupFooter</b>	expression or a function(items) which returns grouping value	
<b>editable</b>	allow cell editing when clicked	false
<b>type</b>	HTML input type for cell editor	text
<b>pattern</b>	HTML validation pattern for input element	
<b>title</b>	input title or a message to show when editing	
<b>listOfValues</b>	array of values to choose from when editing cell value	
<b>class</b>	column class string. Useful to add text-alignment to the column	

## Grid data

Table data can be loaded from:

1. Inline HTML table

2. Dynamic JavaScript array
3. Server-side API endpoint

### Inline data

Simply put data in table HTML in TBODY TR TD elements:

```
<table id="table1" class="table table-hover table-bordered ">
  <thead class="thead-light">
    <tr><th>Code</th><th>Description</th><th>Qty</th><th>Price</th></tr>
  </thead>
  <tbody>
    <tr><td>100</td><td>first description</td><td>1</td><td>30</td></tr>
    <tr><td>200</td><td>other description</td><td>2</td><td>20</td></tr>
    <tr><td>300</td><td>some text</td><td>3</td><td>10</td></tr>
  </tbody>
</table>
```

### Dynamic data

Configuration option “data” allows setting data from JS array:

```
var demodata=[
  ['100','Service title 100', 2, 134],
  ['120','Some stuff', 10, 160],
  ['310','Goods', 5, 2000],
];
var t1=FathGrid("table1",{
  data: demodata
});
```

### Server-side data

To load data from server, set **serverURL** configuration option to string template:

```
t1=FathGrid("table1",{
  serverURL:'https://jsonplaceholder.typicode.com/posts?_page=${page}&_limit=${size}&_sort=${sort}&_order=${order}&q=${search}&${filters}',
```

note that URL string has these placeholders:

- `${size}` replaced with page size setting
- `${page}` page number (1,2,3...)
- `${sort}` sort field names
- `${order}` order of sort, “asc” or “desc” for sort fields
- `${search}` search term (if any)
- `${filters}` replaced with filter name/value pairs (eg: name=test&price=1)

## API

FathGrid object has the following methods:

<b>Method</b>	<b>Parameters</b>	<b>Description</b>
<b>getData</b>		returns read-only copy of table data
<b>setData</b>	<b>data</b> :array	sets new data
<b>export</b>	<b>format</b> : 'txt', 'csv', 'xls', 'pdf'... <b>filename</b> : downloaded file name	downloads data-export file
<b>render</b>		re-draws the table
<b>sort</b>	<b>column_index</b> : 1-based column <b>desc</b> : true for descending order <b>multi</b> : true for multicolumn sort	sort data by field
<b>getSort</b>		returns array of field indices by which data is sorted
<b>setSort</b>	<b>new_sort</b> : array of column indices	sets sort fields
<b>filter</b>	<b>column_index</b> : column <b>value</b> : filter value	sets data filtering
<b>getFilter</b>		returns current filter
<b>editCell</b>	<b>rownum</b> : 1-based row index <b>col</b> : 1-based column index	starts editor in cell
<b>search</b>	<b>q</b> : search term	sets instant search term
<b>getSelectedItem</b>		returns data item from currently selected row
<b>deleteRow</b>	<b>rownum</b> : 1-based row index	deletes a row
<b>insertRow</b>	<b>rownum</b> : 1-based row index <b>item</b> : new item data	insert new row
<b>setServerURL</b>	<b>url</b> : string	set new server URL
<b>wrapperEl</b>		returns FathGrid wrapper HTML Element
<b>selectRow</b>	<b>rownum</b> : 1-based row index	selects specified row

<b>showSubgrid</b>	<b>subgrid:</b> details FathGrid <b>html:</b> custom HTML to add before sub-grid	insert sub grid below currently selected row
--------------------	---	--

## How to...

### Display calculated columns

See column configuration **value** and **html** options.

Example which calculates “item total” from “qty” and “price” columns on each row:

```
var t1=FathGrid("table1",{
...
columns:[
  {
    header:'Item Total',
    editable:false,
    value:(x)=>`${parseFloat(x.qty||0)*parseFloat(x.price||0)}.toFixed(2)}`,
    ...
  }
]
```

### Group by expression

See **groupOn** and **showGroupFooter** configuration options.

Example:

```
//grouping
sortBy:[1],//group on first column
groupOn:item=>`${item[0]}`, //group expression
showGroupFooter:true,

selectColumns:true,
columns:[
  {visible:false}, //invisible grouping column
  {
    groupFooter:(data)=>`${data.length} songs`,
  },
]
```

### Add column footer

See **footer** column configuration option.

Example which adds footer with total sum for “item total” column:

```
var t1=FathGrid("table1",{
```

```

...
columns: [
  {
    header: 'Item Total',
    editable: false,
    value: (x) => `${x.qty*x.price}`,
    footer: (data, el) =>
      data.map(x => (x.qty*x.price)).reduce((x, s) => x+s, 0).toFixed(2)
  },
  ...

```

## Change appearance of some rows

See **rowClass** config option.

Example from songs.html sample page:

```

t1=FathGrid("table1",{
  ...
  rowClass: (item,idx)=>
    item[3]== 'The Beatles'?
      'table-info' :
    (item[3]== "The Rolling Stones"?
      'table-warning':
      ''
    ),
  ...

```

## Editing

If config.editable is true, call **editCell(row,col)** API function to start editor.

Shift+arrow keys move editor across table. Columns with **editable=false** option set are skipped.

Insert new row

Call **insertRow** API function.

Delete row

Call **deleteRow** API function.

## Show sub-grid

First, create second FathGrid object which will hold sub-grid data, then, call **showSubgrid** on main grid to insert details grid under currently selected row.



